Middle-East Journal of Scientific Research 24 (9): 2859-2865, 2016
ISSN 1990-9233
© IDOSI Publications, 2016
DOI: 10.5829/idosi.mejsr.2016.24.09.23937

Secure and Safety Trends Updated in High Performance Computing Using Cloud Computing

[1]*D. Sasikumar and* [2]*S. Saravanakumar*

[1]CSE Department, Bharath University, Chennai, India
[2]CSE Department, Karpagam College of Engg., Coimbatore, India

Abstract: High performance computing (HPC) is a branch of technology that concentrates on combining the power of computing devices.HPC evolved due to increase in demand of processing speed. During pervious era Hipc applications have always required large no of computers interconnected in a network such as cluster. Clusters are difficult to setup and maintain both technically and economically. It becomes easier to deploy hip applications in the cloud without worrying about the costs associated with it. They also give guarantees on the quality of services(Qos).It enables everyday's object to become smarter and interactive. The objective of using cloud based HIPC is to deal with faster processing of huge amount of data and higher throughputs for all multiplicities of information and data. Cloud in HPC reduces the cost of infrastructure and software etc. It provides complete assurance to the data to access independently and safely. Issues in cloud:-Trust, Legal Issues, Confidentiality, Authenticity (Integrity and Completeness), Authorization, Security.

Key words: Large-scale data transfer · Flash crowd · Big data · BitTorrent · p2p overlay · Provisioning · Big data distribution · Cloud storage

INTRODUCTION

Cloud computing takes the technology, services, ``and applications that are similar to those on the Internet and turns them into a self service utility. The use of the word "cloud" makes reference to the two essential concepts:

Abstraction: Cloud computing abstracts the details of system implementation from users and developers. Applications run on physical systems that aren't specified, data is stored in locations that are unknown, administration of systems is outsourced to others and access by users is ubiquitous.

Virtualization: Cloud computing virtualizes systems by pooling and sharing resources. Systems and storage can be provisioned as needed from a centralized infrastructure, costs are assessed on a metered basis, multi-tenancy is enabled and resources are scalable with agility.

Processing large datasets has become crucial in research and business environments. Practitioners demand tools to quickly process increasingly larger amounts of data and businesses demand new solutions for data warehousing and business intelligence. Big data processing engines have experienced a huge growth. One of the main challenges associated with processing large datasets is the vast infrastructure required to store and process the data. Coping with the forecast peak workloads would demand large up-front investments in infrastructure. Cloud computing presents the possibility of having a large-scale on demand infrastructure that accommodates varying workloads. Traditionally, the main technique for data crunching was to move the data to the computational nodes, which were shared. The scale of today's datasets has reverted this trend and led to move the computation to the location where data are stored. This strategy is followed by popular MapReduce implementations (e.g. Hadoop). These systems assume that data is available at the machines that will process it, as data is stored in a distributed file system such as GFS,

Corresponding Author: D. Sasikumar, CSE Department, Bharath university, Chennai, India.

or HDFS. This situation is no longer true for big data deployments on the cloud. Newly provisioned VMs need to contain the data that will be processed. Loughran *et al*. exposed how different flavors of big data processing applications could easily be provisioned in the cloud using automated deployment and configuration tools.

MapReduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks. Users specify the computation in terms of a map and a reduce function and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures and schedules inter-machine communication to make efficient use of the network and disks. Programmers find the system easy to use: more than ten thousand distinct MapReduce programs have been implemented internally at Google over the past four years and an average of one hundred thousand MapReduce jobs are executed on Google's clusters every day, processing a total of more than twenty petabytes of data per day [1].

We describe the design and implementation of a high performance cloud that we have used to archive, analyze and mine large distributed data sets. By a cloud, we mean an infrastructure that provides resources and/or services over the Internet. A storage cloud provides storage services, while a compute cloud provides compute services. We describe the design of the Sector storage cloud and how it provides the storage services required by the Sphere compute cloud. We also describe the programming paradigm supported by the Sphere compute cloud. Sector and Sphere are designed for analyzing large data sets using computer clusters connected with wide area high performance networks (for example, 10+ Gb/s). We describe a distributed data mining application that we have developed using Sector and Sphere. Finally, we describe some experimental studies comparing Sector/Sphere to Hadoop [2].

The Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware and it delivers high aggregate performance to a large number of clients. While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points [3].

Cloud-based MapReduce services process large datasets in the cloud, significantly reducing users' infrastructure requirements. Almost all of these services are cloud-vendor-specific and thus internally designed within their own cloud infrastructures, resulting in two important limitations. First, cloud vendors don't let developers see and evaluate how the MapReduce architecture is managed internally. Second, users can't build their own private cloud-infrastructure-based offerings or use different public cloud infrastructures for deploying MapReduce services. The authors' proposed framework enables the dynamic deployment of a MapReduce service in virtual infrastructures from either public or private cloud providers [4]. In contrast, on a physical cluster Hadoop is already predeployed and the data has been populated beforehand. The task of populating freshly deployed VMs with large chunks of data may look trivial for small datasets, where a simple sequential copy from a central repository to the newly deployed VMs works well. As data size grows (and therefore the number of VMs to process it), data transfer becomes one key performance bottleneck even within the same data centre.

Populating the newly deployed VMs may take prohibitively long (1 TB of data sequentially loaded to 100 VMs on a 10 Gb ethernet network may take 1500 min1). Sometimes this time may be 3-4 orders of magnitude longer than the time it takes to make the actual computations (tens of minutes to crunch data that takes days to load into the VMs). Big Data exposed as a Service (BDaaS) on top of an Infrastructure as a Service (IaaS) cloud is, thus, a complex environment where data distribution plays a crucial role in the overall performance [5].

Background Work: Cloud computing is recognized as an alternative to traditional information technology due to its intrinsic resource-sharing and low-maintenance characteristics. In cloud computing, the cloud service providers (CSPs), such as Amazon, are able to deliver various services to cloud users with the help of powerful datacenters. By migrating local data management systems into cloud servers, users can enjoy high-quality services and save significant investments on their local infrastructures. One of the most fundamental services offered by cloud providers is data storage. Let us consider a practical data application. A company allows its staffs in the same group or department to store and share files in the cloud. By utilizing the cloud, the staffs can be completely released from the troublesome local data storage and maintenance. However, it also poses a significant risk to the confidentiality of those stored files. Specifically, the cloud servers managed by cloud providers are not fully trusted by users while the data files stored in the cloud may be sensitive and confidential, such as business plans. To preserve data privacy, a basic solution is to encrypt data files and then upload the encrypted data into the cloud. Unfortunately, designing an efficient and secure data sharing scheme for groups in the cloud is not an easy task due to the following challenging issues. The correctness of the data in the cloud is being put at risk due to the following reasons. First of all, although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they are still facing the broad range of both internal and external threats for data integrity. Disadvantages: Security is very low, so that the users are afraid of uploading the data in the Cloud Servers. No Proper mechanism was implemented to the audit the data that are stored in the Cloud Servers. As Business point of view the customer's of the Company will be reduced by using this poor Data Auditing Mechanism. We are implementing the secure system namely Privacy preserving auditing. In this system, first the Data Owner will register with the Cloud Service Providers. During the registration phase the Public and Private will be generated for the Data Owner. The Data Owner has to provide their Private Key while updating their data in the Cloud Server. The Cloud Server will allow the Trusted Party Auditor (TPA) to audit the data that was Stored in the Cloud Server as requested by the User. The TPA will also audit multiple Files also. Secure Multi Owner Authentication technique by which we can secure the data Stored in the Cloud Server's Database. First data will be uploaded by the Data Owner in the Cloud Server in the Encrypted format. Also the User wants to View/ Download the data; they have to provide the public key. The Data Owners will check the Public Key entered by the User. If valid, then the decryption key will be provided to the User to encrypt the data. We are also implementing the Load Balancing Concept to Process the User requested Job. First the User request will be past to the Cloud Server and then to the Queues in the Cloud Server. Then the Job will be given to the Virtual Machines presented in the respective Queue By providing the Public and Private key components the user is only allowed to access the data.By allowing the Trusted party Auditor to audit the data will increase the Trustworthiness between the User and Cloud Service Providers. By using Merkle Hash Tree Algorithm the data will be audited via multiple level of batch auditing Process. As Business Point of view, the Company's Customers will be increased due to the Security and Auditing Process.

Big Data as a Service in the Cloud: IaaS cloud providers offer computation and storage resources to third parties [6]. These capabilities are exposed by means of an imperative API that allows customers to deploy VMs based on predefined virtual images, as well as persistent storage services. In addition to the fundamental functionality of providing computing and storage as a service, providers offer an increasing catalogue of services (e.g. authentication and key management), although these do not provide functional building blocks for setting up a generic Big Data Processing infrastructure. Some attempts have been done at setting up Hadoop in the cloud (see [5] and Open Stack Sahara).

In this section we examine the initial provision of a big data service (e.g. Map Reduce or a large scale graph analytics engine) on top of the API exposed by the IaaS provider. Assuming we have predefined VM images containing the required software, we still need to configure the distributed processing platform nodes and provide each node with data for processing. This "data loading" process is often overlooked by most research papers, but it would be essential for a fair comparison on the results obtained in different cloud infrastructures.

The sequence of tasks needed to prepare a big data job for parallel analysis on a set of recently deployed VMs is as follows:

Partitioning: The data set is split and assigned to worker nodes (VMs), so that data processing can occur in parallel.

Data Distribution: Data is distributed to the VM where it is going to be processed.

Application Configuration: The VMs have big data applications correctly configured. Load data in memory. In some computing models, during job preparation, the data must be loaded from the hard disk to RAM.

We discuss each task in the following subsections. We present design alternatives for the tasks, discussing the pros and cons of each design. Discussion is grounded on experimental results and assessment of the manageability of the different alternatives. These preliminary experiments lead the way for us to propose architecture for offering BDaaS on top of a virtualized infrastructure.

Data Partitioning The key for most big data systems is scaling horizontally (or scale out), in the hope that adding more resources (VMs) will reduce the overall execution time. Therefore, partitioning the data and assigning partitions to each VM is the first task of the big data job configuration.

Placing the right data on each VM is an NP-hard problem [7], with different constraints depending on the specific computation model. There are numerous heuristics in the literature that try to find partitioning schemas that can both: 1) keep partitions balanced (so that all the VMs crunch an equivalent portion of the dataset) and 2) reduce the number of messages exchanged between VMs to synchronies their state.

Depending on the selected partitioning heuristic, data partitioning can be a very time consuming task (see [8] for instance).

Data Distribution: Deploying small datasets across a few servers (or VMs) is a trivial task. However, deploying TBs of data across hundreds or thousands of servers becomes substantially more challenging from a performance perspective. Substantial amounts of data must be transferred to each newly created VM2 [9].

Data distribution approaches should focus on reducing the overall transfer time. Reducing the total amount of data transferred around the local network reduces the impact of deploying such a service on the network. However, the cost factor from renting idle VMs waiting for the data transfer is the dominating element in this analysis 3. It must also be considered that a data centre environment provides no guarantees of available traffic, which can render any prediction-based strategies useless. In this subsection we analyses the different approaches for performing data distribution among the deployed VMs.

Design of the Proposed Solution: The rest of the paper is organized as follows. In Section 3 we describe the problem in clouds Virtual machines (VMs) can be provisioned on demand to crunch data after uploading into the VMs. While this task is trivial for a few tens of VMs, it becomes increasingly complex and time consuming. Disadvantages are Poor performance, High time consuming process, less security disadvantages, Privacy less.

User Registration: Here first the User wants to create an account and then only they are allowed to access the Network. Once the User creates an account, they are to login into their account and request the Job from the Cloud Service Provider. Based on the User's request, the Cloud Service Provider will process the User requested Job and respond to them. All the User details will be stored in the Database of the Cloud Service Provider. In this Project, we will design the User Interface Frame to Communicate with the Cloud Server through Network Coding using the programming Languages like Java. By sending the request to Cloud Server Provider, the User can access the requested data if they authenticated by the Cloud Service Provider.

Cloud Server: Cloud Server is the major main server which contains the index data of the entire data present in all sub Cloud Servers. The Cloud Server will act as the main server to receive the query from the user. The main Cloud Server checks the query and match with the index data present in it.

Data Encryption: User will be specifying the index file along with the Data for the easiest process for the data access. Users can easily access the files by searching with some keywords; those keywords are represented as Index Files. So while uploading the file the user will be giving Index files and their Public keys along with the Encrypted File. And this index value, public key and files are encrypted by the AES algorithm and stored in the server.

Data Splitting and Chunking: In this module we are going describe about the chunking, that is chunking mean a large file is split into eight parts of files and they have been stored in different server. The use of splitting is to reduce the load of server and to provide quality of service to the cloud user while they were uploading or downloading files. And another thing we are implements security to the file while we split form main cloud.

ABE Based Data Access: Attribute-based encryption (ABE) can be used for log encryption Instead of encrypting each part of a log with the keys of all recipients; it is possible to encrypt the log only with attributes which match recipients' attributes. This primitive can also be used for broadcast encryption in order to decrease the number of keys used. If a user wants to access any information about any user then he will give his individual key as well as the group key. If he want to access the information about the user, but the user is not belongs to their access policy is not possible. He can only access the user's information within their group only. Without knowledge of the other key it is not possible to access the information.

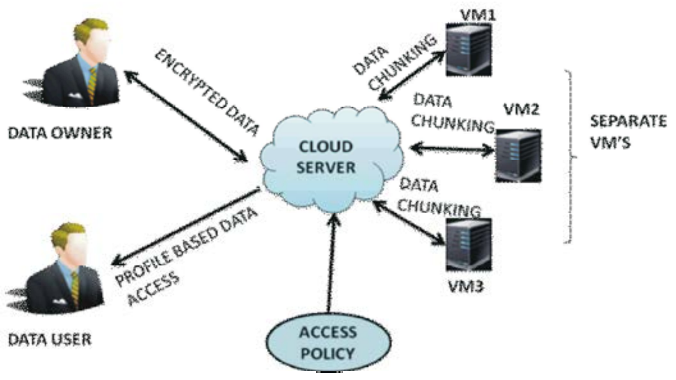

Fig. 3.1: Architecture design

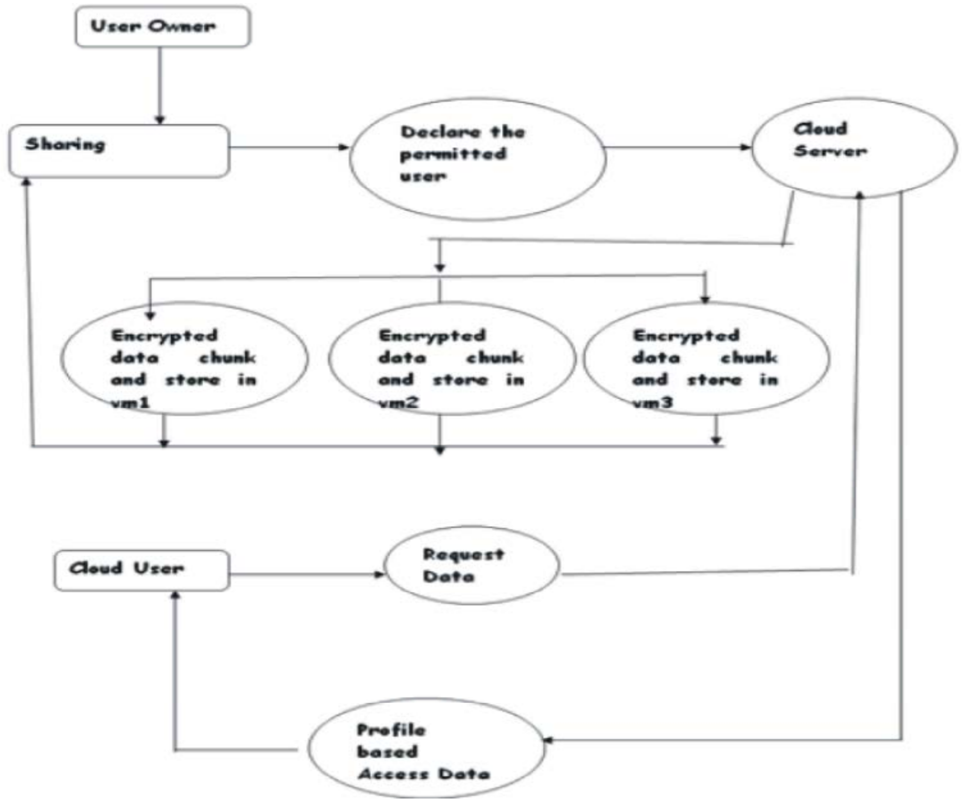
User Owner
Sharing
Declare the permitted user
Cloud Server
Encrypted data chunk and store in vm1
Encrypted data chunk and store in vm2
Encrypted data chunk and store in vm3
Cloud User
Request Data
Profile based Access Data

Fig. 3.2: Data Flow diagram

DISCUSSIONS

Most public cloud vendors do not charge extra fees for intra cloud transfers, but the cost to upload the data to the public cloud can be significant. In our work we assumed the dataset had been pre-uploaded to the public cloud and the population of VM with the data occurs only within the cloud provider's network (no extra charge per new deployment). In case users want to upload their data from their own facilities, they should also attempt to balance the time it takes to upload data to the cloud (possibly over highly asymmetric links) with the amount of money charged by the cloud provider for uploading data into the cloud. The associated costs for downloading the results back to the user's facilities might require similar analysis. Existing big data processing platforms make assumptions on latency and bandwidth that might not hold in a cross data centre environment, severely hurting performance [10]. We have not explored cross data centre configurations of our data distribution approach. We suspect some of the optimizations (low time out times, fewer retrials, agnosticism regarding ISP traffic policies and the like) will not apply in this context and new challenges will be presented. Some authors have successfully tested this idea [11] and we believe some engineering optimization work can make across-data-center transfers more suitable for on demand big data service instantiation than it is today.

While our approach scales well to a few thousand VMs, it could be argued that tracker-fewer approaches are more scalable. That would be a fair statement, but we know of few systems that require more than ten thousand machines today. In addition, our experiments with a DHT client show that the flash crowd effect can be too large when 1, 000 VMs are trying to get chunks of data from a single node. In the future, we would like to improve the interaction between SF and a tracker less Bit Torrent deployment by, for instance, modifying well known protocols to filter unauthorized peers out [12].

We have shown the performance of our system with reasonably big datasets (TBs) and large number of VMs (100 and 1, 000). We did not include results at a smaller scale as they would be less relevant in the context of big data processing. On the other hand, scaling up to 10000 VMs and 1PB would be impractical, as it would take too long to be usable for on-demand data analytic tasks in the cloud (further advances in data centre networking, e.g. optical connectivity to the rack, are required). The limits on data size can also be circumvented using complementary techniques. For instance, many data analytics applications work on text files and very good compression rates are easy to obtain (5:1 is not at all uncommon, even more in structured logs). Thus, a 100 TB file could be reduced to roughly 20 TB, which still would not fit in our 100 VMs and would take a prohibitive amount of time unless proper pre partitioning and wiser transfers were scheduled. Highly consolidated physical infrastructures imply VMs share room with many other neighbors. Some of these tenants may be less desirable roommates, especially those that make heavy use of the shared IO resources. Thus, VM allocation can cause significant variability on the obtained results. Smarter scheduling policies taking big data applications into account are needed. The academic community has proposed several alternatives for this type of smarter scheduling (see [13], [14] for instance), but few of these works have been tested in large scale environments, which makes transfer into products more complicated.

Object stores such as Open Stack's Swift could be used as a first hop (instead of having our initial relay nodes). This is conceptually analogous to our approach. Assuming the object store nodes would play the role of our relay nodes, then our practitioner would need to upload the initial tor- rents to separate buckets in the object store (as it happens now with the relay VMs).

Debugging problems across logs distributed in thousands of nodes is a labour intensive problem. Being able to re-run a trace of all the events that occurred that lead to the failure/problem is essential to be able to reproduce the problem and fix it. Having smarter schedulers may pay off (see two paragraphs above), but not at the expense of obscuring devops-level understanding.

Uplink utilization is critical for the performance of BitTorrent clients in classic scenarios. Every 10 seconds a node "unchokes" the k ¼ 4 default nodes which have pro- vided it with the highest download rates during the previous 20 sec. Each of the unchoked nodes is thus given an equal probability of pulling the missing chunks of data. Some works have reported 90 percent utilisation of the uplink except for the initial period where utilisation is very low. Some authors have tried to use new protocols to improve the utilisation of the uplink [15]. Our technique is not constrained by limited asymmetric DSL uplink connectivity and therefore, uplink capacity is not our main concern. We limit the maximum transfer rates, though, so that we ensure peers do not consume all the network capacity in our data transfers.

Our approach works well in practice partly thanks to the enforced coordination at the initial stages. Network coding promises optimal utilisation of the available bandwidth [16]. Future work aims to explore the introduction of net- work coding mechanisms as another tool for the partitioner to schedule transfers of data to the right VMs.

CONCLUSIONS

Provisioning thousands of VMs with datasets to be crunched by their big data applications is a non trivial problem. A big data provisioning service has been presented that incorporates hierarchical and peer-to-peer data distribution techniques to speed up data loading into the VMs used for data processing. The method is based on a modified BitTorrent client that is dynamically configured by the software provisioning modules. Peers are initially configured in a tree topology, where a subset of VMs plays the role of relay nodes (preventing flash crowd effects on the permanent data storage). As soon as some data chunks start to be ready in the leaves of the tree, the topology evolves to a classic P2P mesh shape. Our implementation and evaluation with hundreds of TB and thousands of VMs show this is an effective method for speeding up dynamic provision of big data applications in the cloud, obtaining improvements on transfer times around 30 percent over current state of the art techniques. This may represent significant savings in the price paid by users of public clouds. At the same time, our system keeps a low entry barrier for users who may not be experts in infrastructure management (they deal with a single high-level declarative configuration file and the system takes care of configuring software and data loading).

REFERENCES

1. Dean and S. Ghemawat, 2008. Mapreduce: Simplified data processing on large clusters. Commun. ACM, 51(1): 107-113. [Online]. Available: http://doi.acm.org/10.1145/ 1327452.1327492.
2. Grossman, R. and Y. Gu, 2008. Data mining using high performance data clouds: Experimental studies using sector and sphere. in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '08, New York, NY, USA: ACM, pp: 920-927. [Online]. Available: http://doi.acm. org/10.1145/1401890.1402 000.
3. Ghemawat, S., H. Gobioff and S.T. Leung, 2003. the google file system. in Proceedings of the 19th ACM Symposium on Operating System Principles, ser. SOSP '03, New York, NY, USA: ACM, pp: 29-43. [Online]. Available: http://doi.acm.org/10.1145/ 945445.945450.
4. Loughran, S., J. Alcaraz Calero, A. Farrell, J. Kirschnick and J. Guijarro, 2012. "Dynamic cloud deployment of a mapreduce architecture, " Internet Computing, IEEE, 16(6): 40-50.
5. Vaquero, L.M., A. Celorio and F. Cuadrado, 2015. "Deploying large-scale datasets on-demand in the cloud: Treats and tricks on data distribution," - IEEE Transactions on …, 2015 - ieeexplore.ieee.org.
6. Vaquero, L.M., L. Rodero-Merino, J. Caceres and M. Lindner, 2008. "A break in the clouds: Towards a cloud definition, " SIGCOMM Comput. Commun. Rev., 39(1): 50-55, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496100.
7. Vaquero, L.M., A. Celorio and F. Cuadrado, 2015. Deploying large-scale datasets on-demand in the cloud: Treats and tricks on data distribution - IEEE Transactions on …, 2015 - ieeexplore.ieee.org.
8. Celorio, Y.A. and R. Cuevas, Deploying Large-Scale Data Sets on-Demand in the Cloud: Treats and Tricks on Data Distribution. - pdfs.semanticscholar.org.
9. DM Meisner - Architecting Efficient Data Centers - 2012 - deepblue.lib.umich.edu.
10. Vaquero, L.M., A. Celorio and F. Cuadrado, 2015. Deploying large-scale datasets on-demand in the cloud: Treats and tricks on data distribution … Transactions on Cloud …, 2015 - ieeexplore.ieee.org.
11. Cuevas, R., N. Laoutais, X. Yang, G. Siganos and P. Rodriguez, 2013. "Bittorrent locality and transit traffic reduction: When, why and at what cost?" Parallel and Distributed Systems, IEEE Transactions on, 99: 1-1.
12. Rasti, Ekbatani H., 2013. Investigating the Mutual Impact of the P2P Overlay and the AS-level Underlay-2013 - scholarsbank.uoregon.edu.
13. Faber, M.H. and M.G. Stewart, 2003. Risk assessment for civil engineering facilities: critical overview and discussion - Reliability Engineering & System Safety, 2003 – Elsevier.
14. Datta, R., D. Joshi, J. Li and J.Z. Wang, 2008. Image retrieval: Ideas, influences and trends of the new age-ACM Computing Surveys (CSUR), 2008 - dl.acm.org.
15. Dhiman, A., 1956. Vertical and horizontal handover in heterogeneous wireless networks-1956 - dspace.thapar.edu.
16. Alvaro, P., T. Condie, N. Conway and K. Elmeleegy, 2010. Boom analytics: exploring data-centric, declarative programming for the cloud- Proceedings of the 5th …, 2010 - dl.acm.org.